Visualization of On-demand Virtual Data Products in a Distributed Environment

Mike Botts (mike.botts@atmos.uah.edu)
Earth System Science Center [11]
Global Hydrology and Climate Center [3]

Ken Keiser (kkeiser@itsc.uah.edu)
Helen Conover (hconover@itsc.uah.edu)
Sara Graves (sgraves@itsc.uah.edu)
Information Technology and Systems Center [10]

University of Alabama in Huntsville
Huntsville, AL, 35899

The Information Technology and Systems Center (ITSC) and Earth System Science Center's VisAnalysis Systems Technologies (VAST) Team, both at the University of Alabama in Huntsville (UAH), have collaborated on an information system that provides the ability to access, subset, customize, and visualize data across a distributed system of data sources. This work has been funded, in part, by NASA's Earth Science Information Partnership (ESIP) Federation, and the effort has been targeted towards Earth Science data sets. These UAH research groups have been the primary information system participants in the Passive Microwave ESIP-2 (PM-ESIP) project.

The working prototype ESIP Federation [2] is an enterprise model to facilitate the public availability of data from heterogeneous, geographically dispersed providers. The PM-ESIP project is prototyping new information system approaches to providing users advanced ways to find, view and retrieve Earth Science data sets. Some of the advances of the PM-ESIP prototype include real-time access to distributed, heterogeneous data sets, use of sensor models for the resolution of temporal and spatial parameters and for on-demand geolocation, interactive visualization of data (not just pictures) across the network, real-time subsetting of data based on user-defined parameters, and use of advanced client-server technologies for the communication of requests and resulting data.

Many data centers are being faced with the issues involved with maintaining the large amounts of Earth Science data now being collected daily. Large data sets are maintained at various distributed locations and it is not always feasible to "copy" all the data a user might want to analyze or view. Due to the size of some of these data sets, it is not always practical to create and store a variety of derived data product files on server disks when the system is capable of creating the virtual data "on-demand" for delivery to users. The PM-ESIP team has been investigating possible solutions to some of these data issues.

2 PM-ESIP System components

The product on-demand portion of the PM-ESIP information system is composed of two main components. The visualization client is a customization of the Space-Time Toolkit (STT) application developed by the VAST team at UAH. The STT is able to retrieve data dynamically from distributed server platforms and display multiple inputs simultaneously in geographical space, based on a specified temporal parameters. The production of data on-demand is handled by the Algorithm Development and Mining (ADaM) system, developed by the ITSC team, also at UAH. ADaM is an advanced data processing system that provides support for easily incorporating many different input/output data formats and a framework for adding customized scientific algorithms for extensible data analysis capabilities.

These two components of the PM-ESIP system had originally been developed independently by their respective research centers, prior to the ESIP collaboration. The flexible architectures of the STT and ADaM afforded the centers the opportunity to investigate the use of some new technologies to tie the components together in order to provide new and complementary functionality that was not possible with the separate pieces. The STT client provided a visualization interface into the data products resulting from the ADaM processing, while ADaM was able to broaden the range of data products and capabilities available to STT users.

2.1 Space Time Toolkit (STT)

The UAH VAST team has been involved in the development of two advanced technologies, the Observation Dynamics Model (ODM) and Space-Time Toolkit (STT), that provide on-demand geolocation and processing of low-level sensor data, and on-the-fly integration of disparate data sets within a highly interactive visual environment, respectively [1].

2.1.1 Sensor Modeling and On-Demand Georectification

Through the development and application of Observation Dynamics Model (ODM) software, the UAH/VAST team has demonstrated the ability to geolocate virtually any dynamic sensor system, on an “as needed” basis within an interactive desktop environment. These sensor systems, for example, can include virtually any scanner, imager, or profiler carried by satellite, aircraft, balloon, ship, land-vehicle, or at a stationary site (e.g. Doppler radar or video camera), as well as any virtual camera defined by the user to render a display scene. The ODM is based on the principle of providing a general library for transforming the sensor system data through appropriate translations, rotations, and projections, and then providing sensor system specifics through the use of various descriptor files (e.g. platform position, attitude, planet model, sensor model, etc.). It has been demonstrated that the ODM transforms are sufficiently fast to allow on-the-fly geolocation and transformation of low-level sensor data as needed within a laptop visualization or analysis environment.

2.1.2 Visual Integration of Disparate Data

In addition to the ODM efforts, the UAH/VAST team has been developing advanced technologies for improving the visual integration and analytical comparison of disparate data sets. The integration of multiple data sets is often very challenging because of disparities in the spatial and temporal representation of these data. For example, typical environmental data sets for atmospheric conditions might include raw sensor data from polar orbiting or geostationary satellites, radial scans from a Doppler radar, 2D and 3D gridded map-projected products derived from models or sensor data, multivariable data collected at various times from irregularly spaced stations, vertical profiles derived from sounders or rawinsondes, measured or derived data along aircraft paths or paths of hurricanes, irregular event data (e.g. lightning strikes), polygonal or vector data derived from Geographical Information Systems (GIS), and temporal-spatial annotations.

The Java-based Space-Time Toolkit (STT) allows visual integration of spatially and temporally disparate data sets within a highly interactive, user-driven environment. The STT is capable of visually integrating virtually any geographic data sets, regardless of disparities in their spatial and temporal representation. This includes the integration of low-level sensor data, gridded products, vector based GIS data, and 3D model output, to name a few. The STT is particularly adept at effectively integrating highly dynamic data. A key component of the STT is the use of the lightweight ODM described above for on-demand geolocation and processing of low-level sensor data. Unlike most tools that require that data be converted to a common spatial and temporal grid before integration, the STT allows one to ingest swath, map-projected, station, event, path, or GIS vector-based data in whatever spatial and temporal domain in which it exists. The STT allows the end-user to select any relevant 2D or 3D display domain and then on-the-fly dynamically maps all data into that domain as needed.

In addition to the user-selection of the spatial domain, the end-user has equal control over the aggregation and resolution of the temporal domain. Within the STT, time is treated as a continuum along which data attaches itself when updates are available. As the user scrolls through time (or as time advances in a real-time application), data objects immediately update themselves as appropriate. This allows data sets with different temporal resolutions to still be synchronized, as well as co-registered with other disparate data sets. The time step, the persistence of each data set, and the lag or precession of individual data sets relative to the master time are all user-selectable. Furthermore, by altering the persistence for any particular data item, the user can control how long that data item will remain in the display once it becomes active. This effectively translates into a user driven ability to mosaic or collect data over a dynamic time range.

The STT has proven itself useful for highly interactive visualization and exploration of disparate data sets, as well as for client-side coincidence search of available dynamic data. The application of more intelligent software on the client side allows a more effective use of data transfer bandwidth by limiting requested information to only the information relevant to the user's needs. In addition, the retention of data in its original temporal and spatial state within the STT provides significant advantages for adding sophisticated capabilities for data processing, analysis, and fusion.

The use of Java as the main programming language for STT has many benefits, including the ease of incorporating "web-smarts" into the STT. The STT can access data through URL links, Java Servlets, Java Database Connectivity, or locally on disk. STT also has build-in support for the eXtensible Markup Language (XML), which provides an increasingly popular means for distributing both data and metadata across the web. In addition, the STT can be run as an applet within a web browser or as a standalone web-smart application with the same level of functionality.

The STT originally focused on accessing distributed data through the use of URLs which pointed to remote data files. The data was then accessed through Java I/O streams, requiring the STT client to understand the format and structure of the data files. More recently, STT development began moving more toward data access through the use of server-side Java Servlets. These provide significant benefits, including (1) greater flexibility and easy extensibility for providing an array of services, (2) a reduction in data volume being sent over the web, (3) avoidance of issues regarding the use of native code (i.e. C++ or FORTRAN) on the client, and (4) much lighter-weight client-side code by moving more data expertise to the server.

2.1.3 Application of the STT and ODM

The utilization of the ODM within the STT has provided significant capabilities. One is the ability to synchronize, co-register, and transform various sensor data into any user-defined display space. The example STT display in Figure 1, illustrates the visual integration of Advanced Microwave Precipitation Radiometer (AMPR) sensor data, flying onboard an ER2 aircraft, with several elevations of reflectivity data from two Doppler radars. On-the-fly geolocation of these sensors using the ODM and sensor models has proven faster than reading and using pre-calculated latitude-longitude-altitude values for each pixel.

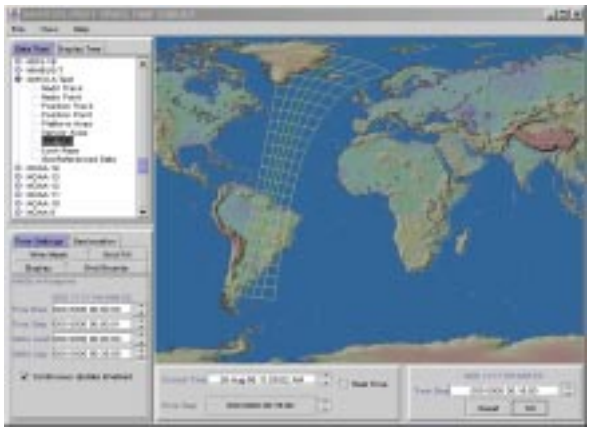

Figure 1-A

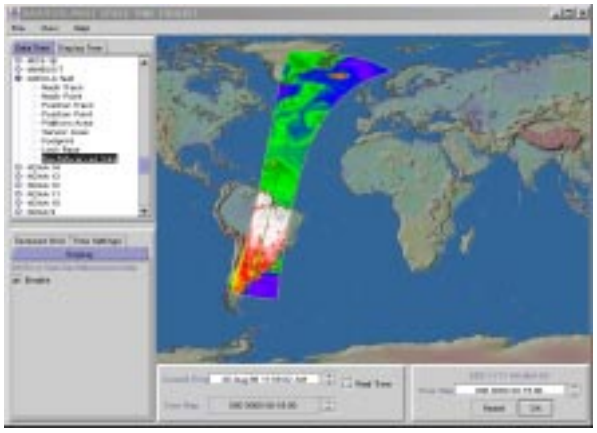

Figure 1-B

Space-Time Toolkit display (1-A) showing the footprint location of the AMSU-A sensor for a given time range and the display of the sensor data (1-B), after on-demand subsetting, processing, and delivery over the web by ADaM. The ODM within STT allows determination of footprint location and provides on-the-fly geolocation of the sensor data into the display.

An added benefit of utilizing an ODM is that the position, look angles, and footprint of any sensor system can be determined for any time. This allows one to search for coincidence between multiple sensors or between any given sensor and a particular event of interest (e.g. hurricane, severe storm, or earthquake). The STT uses web-based retrieval of ODM parameters (e.g. platform positions and sensor parameters) to derive the movement and geolocation of any of a large number of sensor systems, as shown for AMSU-A in Figure 1-A. When one determines that the sensor is looking at an area of interest, one can request the actual data. As part of the NASA PM-ESIP research grant, the UAH VAST and ITSC teams collaborated to develop Java Servlets capable of receiving a web-based request from the STT, subsetting the appropriate swath data set, and returning that subsetted data to STT for immediate display (Figure 1-B). The ability to request the highest resolution data for only the limited area of interest is vital to keeping the size of the transferred data to a small volume such that on-demand subsetting, distribution over the web, and on-the-fly application of algorithms are reasonable.

For more information on the Space Time Toolkit and other UAH VAST activities, visit http://vast.uah.edu. The Space Time Toolkit can be downloaded and installed at http://vast.uah.edu/SpaceTimeToolkit.

2.2 Algorithm Development and Mining (ADaM)

The ITSC originally developed the Algorithm Development and Mining (ADaM) data processing system under a research grant from NASA to investigate new methods of processing large volumes of Earth Observing System (EOS) remote sensing data sets, such as data mining [5]. The intent with ADaM was to provide a data processing environment that could help insulate researchers from the overhead of dealing with the myriad data formats prevalent in the Earth Science community, and at the same time provide researchers the ability to easily extend and add new data processing steps through the introduction of new algorithms, in the form of ADaM operations. The realization of these features has resulted in a system that has proven to be versatile in applying scientific analyses to large and varied scientific data sets. The ADaM system has subsequently been used for research studies dealing with topics such as data mining, texture classification, image processing, subsetting, and statistical analysis of earth science data sets [6, 9].

2.2.1 ADaM Processing Architecture

The nature of earth science spatial data is that it tends to come it many shapes and sizes in terms of formats, scales, resolution, etc. This has historically caused time-consuming problems when researchers attempt to integrate new data sets into studies and models. In an attempt to alleviate some of these problems, ADaM was designed in such a way as to internally handle all data in a common medium regardless of the actual input and resulting output formats. This approach effectively insulates the internal analysis modules from data set specific concerns, thereby simplifying the development of those modules.

It is frequently the case that multiple distinct steps are required to accomplish a given processing task. In order to perform this processing without writing intermediate results to long term storage (disk),ADaM's designers took a data pipeline approach. Results from one processing operation form the input to the next operation, and any number of operations may be chained together in a single processing plan. Figure 2 illustrates the functional architecture of the ADaM engine.

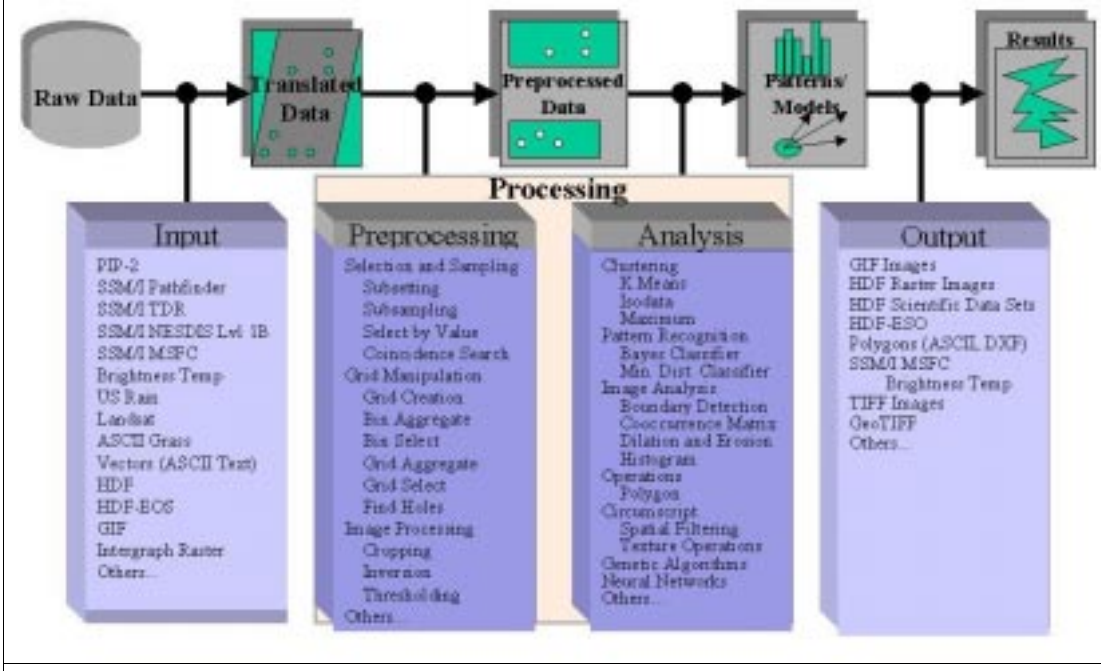

Figure 2: ADaM Engine Architecture

2.2.2 ADaM Design Goals

ITSC developers targeted the use of some of the latest software design concepts for the ADaM system in order to achieve the highest degree of portability, accessibility and modularity for the system. The core, or engine, of the system was developed using standard C++, avoiding unnecessary system calls, thereby making the software compatible with running on multiple operating systems including the most common UNIX flavors, as well as Microsoft Windows NT. The system also needed to be capable of operating in environments ranging from a data archive center to a user's desktop workstation. Distributed access to the data processing functionality was required to deliver end user applications via the web and on standalone workstations. Network accessibility was achieved by use of conventional socket communications originally, and most recently by the development of an ADaMServer component that utilizes Java Servlet technology to handle requests from client applications. The servlet technology utilizes typical HTTP communications to transfer requests and responses, which provides a certain level of security to the client-server communications that was not complete in

the daemon approach. Code modularity was required to allow for extensibility and modification of the operations/algorithms available for use by researchers. The C++ implementation takes full advantage of object oriented design methodologies. The resulting structure of the system has allowed for easy reuse of software modules throughout the system as well as the easy extension of existing classes for enhanced functionality. The software modules (input filters, processing modules and output filters) communicate with one another through a simple set of messages. These messages insulate the modules from one another and eliminate unnecessary coupling.

For more information on the ADaM data processing system, and other UAH ITSC activities, visit http://itsc.uah.edu/ and http://datamining.itsc.uah.edu/.

3 Integrating STT and ADaM for On-demand Processing

A goal of the PM-ESIP project was to create an environment where users could interactively access heterogeneous custom data products from distributed locations for viewing and ultimate retrieval. The PM-ESIP approach was to take the processing to the data, rather than bringing the data to the processing. In other words, the server side processing elements of the system, ADaM and ADaMServer, are constructed to be installed and configured on a data provider's site (figure 3). The ADaM data processing system provides the means to easily plug in virtually any operation required for reading or writing new data types, and the processing and generation of new data products. The ADaMServer component provides the network communication link necessary for multiple client applications to access the data processing functionality remotely. The Java-based interface of the STT provides an expandable visualization framework in which new data sets from multiple distributed data providers can easily be configured for simultaneous user selection, visualization and analysis. The STT client provides a rich set of geographic display tools including on-the-fly re-projection of map data to different coordinate systems, dynamic re-symbolization of features, interactive pan, zoom and rotation in three dimensions, and interactive and automated temporal incrementing of the visual data. The distinction between the presentation of static data sets and virtual ones generated "on-demand" is hidden from the user on the interface. Furthermore, depending on the size of the data sets being processed or the complexity of the process, the time required to generate, transfer and display the requested data is so minimal that it is typical indiscernible whether the data product existed previously or was generated on-demand.

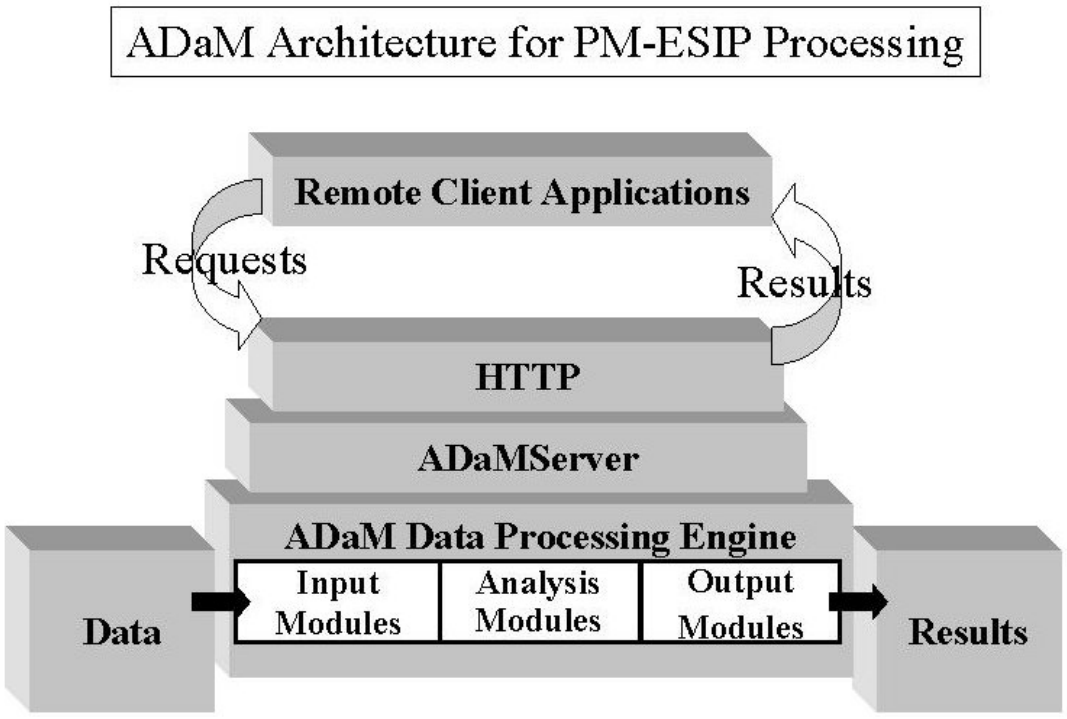

Figure 3: ADaM PM-ESIP Processing Architecture

3.1 Possible User Scenario

The following steps attempt to illustrate how data processing requests and results flow through the integrated PM-ESIP system (figure 4) in response to user interactions.

- Upon startup, the STT client application determines what data resources are available at each of the identified data servers, and builds and displays the appropriate data set list for user selection.
- A user selects a data set from the provided list and indicates a time range of interest on the STT client interface.
- The STT client routes a request for a data subset (temporal and/or spectral) to the appropriate data server.
- The data server retrieves data file(s) and performs subsetting, fusion, or custom processing as required to generate the requested data product.
- The data server sends the requested data back to the STT client in the form of a serialized Java object.
- The STT client receives the serialized data object and displays the data in the user's current view that can be customized in terms of zoom, projection, 3d perspective, etc.
- The user may choose to step the display through time increments. Each step increment results in additional data being requested, processed and retrieved for display.

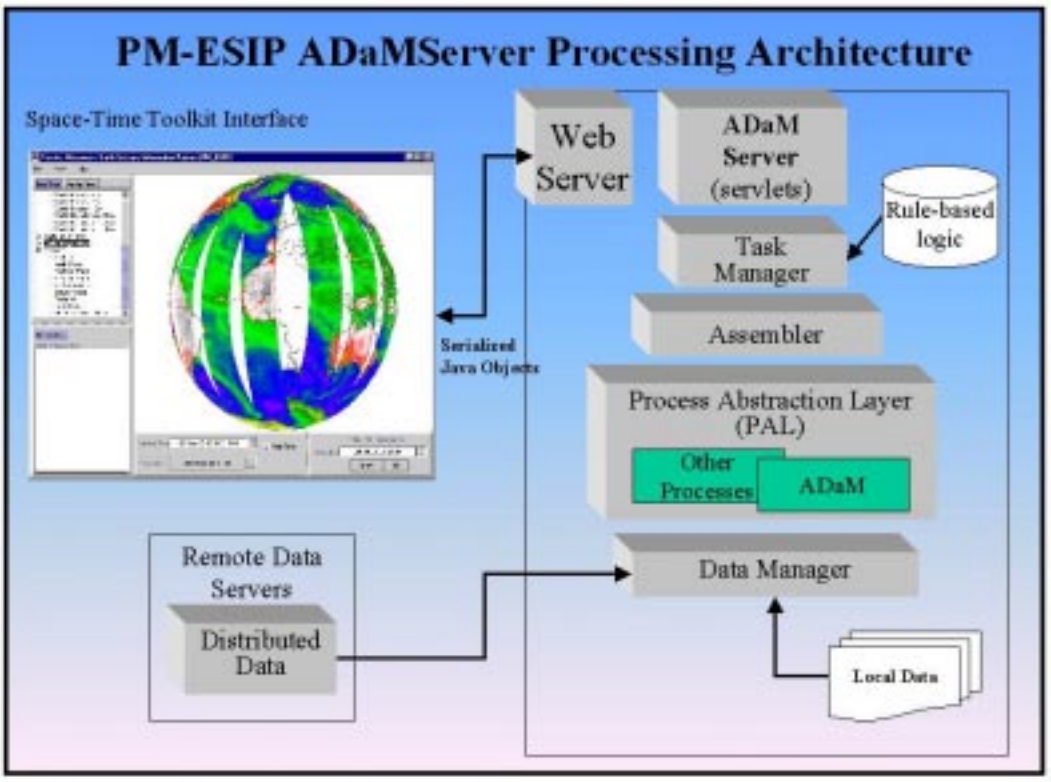

Figure 4: PM-ESIP ADaMServer Processing Architecture

3.2 How PM-ESIP Components Communicate and Why

The PM-ESIP system's client-server communications are implemented using Java servlet technology. The client interface loads resource configuration information at startup that configures the client to send requests for specific data sets to correctly associated data servers. The resource configuration information itself resides on a known PM-ESIP data server and is routinely updated as new data is introduced to the data servers or as new servers are brought online. Through the servlet communication layer, the STT interface is able to pass data requests to any of several data server installations, as shown in Figure 5 below. When the data processing is complete on the data server the results are then passed back to the original client. Both data processing requests and the resulting data are transferred as Java serialized objects. Object serialization is explained in section 3.2.3, but the main benefit of this approach was to provide cross-platform data interoperability between the client and server portions of the PM-ESIP system. This approach, for instance, allows a PM-ESIP client to simultaneously communicate requests to, and receive results from, data servers implemented on different operating systems and hardware platforms.

Servlet technology is essentially Java's answer to the conventional Common Gateway Interface (CGI) programming typically developed in support of dynamic web page generation and servicing. There are many publicized reasons attesting to the benefits of

using Java servlets over conventional CGI scripts, including better cross-platform portability, state persistence across requests, and efficiency. The ITSC-VAST team chose the servlet approach primarily for the portability considerations, the well defined object structure of the servlet classes in Java, and the ability to make use of object serialization for information being passed between the client and server portions of the system.

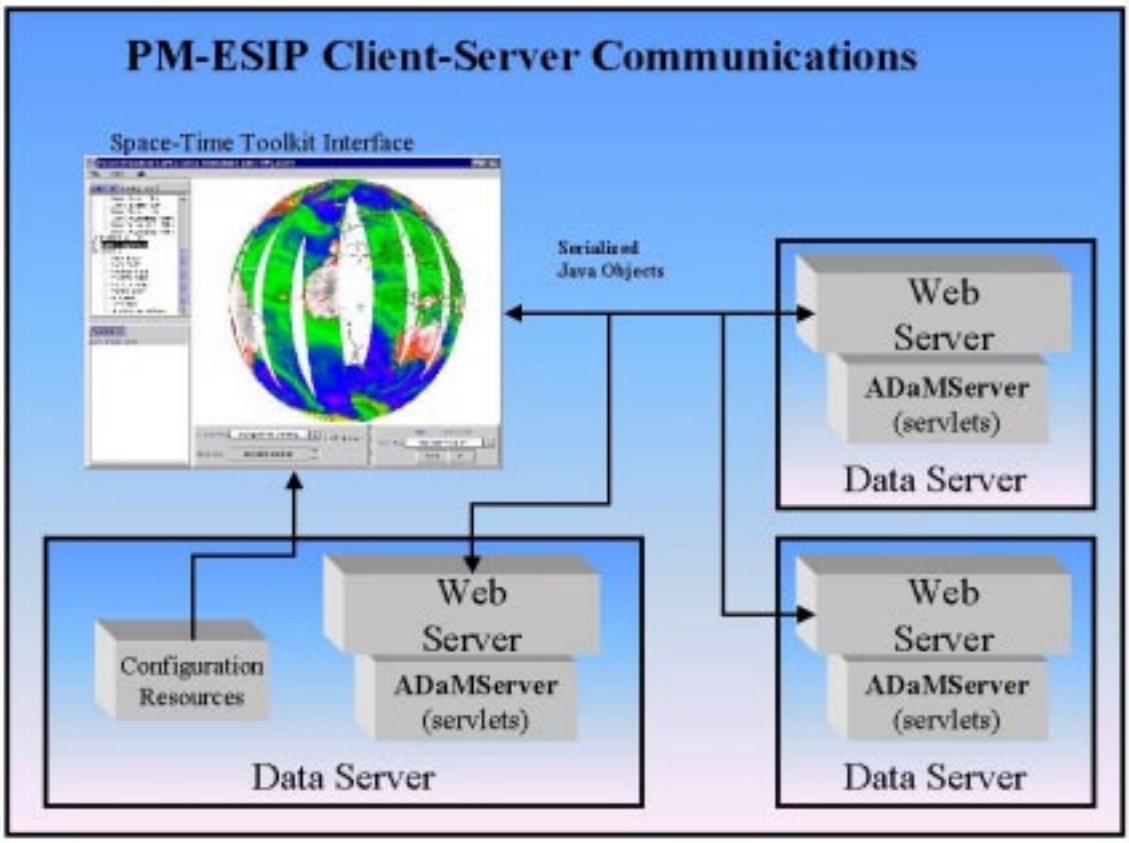

Figure 5: PM-ESIP Client-Server Communications Illustration.

3.2.1 Portability

A key consideration for the system development from both the ITSC and VAST teams has continued to be that of code portability and reuse. With the technologies available today, reasons for writing non-standard or platform-specific code modules are disappearing and are increasingly impractical. Java, by definition, is a portable computer language and has proven to inherently support these claims. There are still instances where a particular vendor's version of the Java virtual machine produces some unexpected results, particularly in the browser-wars, but overall the consistency across like versions of virtual machines is fairly dependable.

3.2.2 Servlet Classes

Since version 1.1 of Java, there has been available a robust and extensible set of classes that handles the HTTP "Get" and "Post" request and subsequent response conventions [4]. The availability and support of this class structure has proven to make servlet implementation a relatively easy undertaking so developers can concentrate more on the actual processing that needs to be handled between receiving a request and generating a response.

3.2.3 Object Serialization

Again, since version 1.1 of Java, the language has supported the concept of object serialization [4]. Serialization provides the ability to store or persist the complete state of an object. This persisted state can be written to a file, or in the case of the PM-ESIP effort, to an output data stream. The result of this approach allows this system to transfer entire objects, containing processing requests or resulting data, between the client and server portions of the system. In this case, since both the client and server portions of the system are written in Java and understand the definition of the objects involved, the interpretation is automatic at either end, making the implementation of client-server communications equivalent to receiving an object through any other method invocation.

4 State of PM-ESIP System

The first public release of the PM-ESIP's interactive product on-demand delivery system is planned for summer 2000, with enhancements expected throughout the following year. Users will be able to access the system from the PM-ESIP web site, http://pm-esip.msfc.nasa.gov.

In addition to the integration effort described above, extending the capabilities of both STT and ADaM to develop a new information system, the PM-ESIP is collaborating with other data providers to further interoperability among systems. For example, during the past year the Open GIS Consortium (OGC), in collaboration with other agencies, has defined a specification for the Web Mapping Testbed [7] that provides a truly platform-independent architecture for passing geographic information to web-based client applications. Although the OGC's first Web Mapping Testbed specification primarily deals with serving map pictures across the network, the basic architecture of the PM-ESIP system is very similar. The PM-ESIP system provides more advanced capabilities for serving actual data to the client and it is expected at the time of this writing that the OGC is headed that direction with future WMT revisions. Plans are underway to add support for the WMT protocol to the PM-ESIP system. While this will not increase the functionality of the PM-ESIP's stand-alone system, it will provide WMT connectivity to other systems being developed at other sites. The PM-ESIP team fully supports the WMT effort and will continue to incorporate developments to the WMT specifications in future development of this information system.

In an effort to extend this functionality to include Earth Science data sets outside the PM-ESIP environment, the UAH team will be configuring deliverable packages of the server portion of this system that can be installed and configured by other data providers.

5 References

[1] Botts and Phillips, 1996

[2] ESIP Federation Web Page, http://www.esipfed.org

[3] Global Hydrology and Climate Center, http://www.ghcc.nasa.gov/

[4] Goodwill, James, 1999. "Developing Java Servlets", Macmillan Computer Publishing

[5] Hinke, T., J. Rushing, S. Kansal, S. Graves, and H. Ranganath, 1997: "For Scientific Data Discovery: Why Can't the Archive be More Like the Web," *Proc. 9th Int. Conf. on Scientific Database Management*, Olympia, WA, Aug. 11-13, 1997.

[6] Keiser, K., J. Rushing, H. Conover and S. Graves, 1999. Data Mining System Toolkit for Earth Science Data. *Earth Observation and Geo-Spatial Web and Internet Workshop (EOGEO)-1999*, Washington, Feb 9-11.

[7] OpenGIS "Web Map Sever Interface Specification version 0.9.9", January 17, 2000.

[8] PM-ESIP Web Page, http://pm-esip.msfc.nasa.gov/

[9] Ramachandran R. , Helen Conover, Sara Graves and Ken Keiser, 2000. "Algorithm Development and Mining (ADaM) System for Earth Science Applications", *Second Conference on Artificial Intelligence, 80th AMS Annual Meeting*, Long Beach, January, 2000

[10] UAH Information Technology and Systems Center, http://www.itsc.uah.edu/

[11] UAH VisAnalysis Systems Technologies – Earth System Science Laboratory, http://vast.uah.edu/